第11回IEEE広島支部学生シンポジウム予稿集
2009/11/21-22 山口大学工学部

モバイルエージェントの移動における トラフィック削減のためのコード転送方式

A Code Transfer Method for Traffic Reduction in Migration of Mobile Agent

東野 正幸 † 本村 真一 ††
Masayuki HIGASHINO† Shinichi MOTOMURA††
笹間 俊彦 † 川村 尚生 † 菅原 一孔 †
Toshihiko SASAMA† Takao KAWAMURA† Kazunori SUGAHARA†
† 鳥取大学大学院 工学研究科 情報エレクトロニクス専攻 †† 鳥取大学 総合メディア基盤センター

1 はじめに

モバイルエージェントとは，ネットワークに接続されたコンピュータ間を移動しながら自律的に処理を行うプログラムである．

モバイルエージェントがコンピュータ間を移動する際は，移動元で行っていた処理を中断して移動先で再開できるように，モバイルエージェントの実行時状態と，モバイルエージェントが行う処理を記述したプログラムコードを，移動先へ転送する必要がある[1]．

このことから，モバイルエージェントが行う処理が複雑になるほど，プログラムコードの量が増大するため，転送に要するトラフィックが増大し，モバイルエージェントの移動に時間がかかる問題がある．

また，モバイルエージェント技術を用いたアプリケーションにおいては，モバイルエージェントの移動に要する時間がオーバーヘッドとなり，アプリケーション全体のパフォーマンスを低下させてしまう問題がある．

そこで本論文では，モバイルエージェント技術を用いたアプリケーションにおいて，モバイルエージェントがコンピュータ間を往復や巡回する状況が多いことに着目し，モバイルエージェントが移動する際に転送したプログラムコードを各コンピュータにキャッシュすることでトラフィックを削減し，モバイルエージェントの高速な移動を実現するプログラムコード転送方式を提案する．また，提案方式を実装して評価した結果，モバイルエージェントの移動を高速化できることを確認したとともに，モバイルエージェント技術を用いたアプリケーションでのパフォーマンスを改善できることを確認した．

本論文では，第2節で従来から提案されているプログラムコードの転送方式の特徴とその課題について述べ，第3節で提案する転送方式の詳細について述べる．第4節では提案方式の実装方法について述べ，第5節では提案方式の評価を行う．そして最後の第6節で本論文のまとめを述べる．

[1]本論文では特に断らない限り，単に「実行時状態」と書いた場合は，モバイルエージェントの実行状態を指し，単に「プログラムコード」と書いた場合は，モバイルエージェントが行う処理を記述したプログラムコードを指す．

2 プログラムコードの転送方式

従来から提案されているプログラムコードの転送方式は，主に以下の3種類に分類できる [1]．この節では，それぞれの概要とその課題について述べる．

コンピュータに予め配置しておく方式 モバイルエージェントが移動する全てのコンピュータへプログラムコードを予め配置しておく方式である．この方式は，モバイルエージェントが移動する際にプログラムコードを転送する必要が無く，実行時状態だけの転送で済むため，モバイルエージェントの移動は非常に高速である．

しかしこの方式は，プログラムコードが配置されていないコンピュータに対してモバイルエージェントが移動した場合に，移動前に行っていた処理を移動先で継続できない問題がある．

必要に応じて転送する方式 モバイルエージェントが移動する際に，まずモバイルエージェントの実行時状態を先に転送し，移動先で実際にプログラムコードが必要になった時点で，プログラムコードを転送する方式である．この方式は，実際に必要なプログラムコードだけを転送するため，転送量を最小に押さえることができる．さらに，モバイルエージェントが移動先で処理を開始するまでのレスポンスタイムも優れている．しかし，コンピュータのネットワークが常時接続でなければならないため，移動元と移動先での非同期実行というモバイルエージェントの利点のひとつが失われてしまう問題がある．

一括して転送する方式 モバイルエージェントの実行時状態と，処理に必要とする全てのプログラムコードを1つにパッケージングして転送する方式である．この方式は，移動時に一度だけの通信で済むため，非常時接続環境で優位となる．

しかし，モバイルエージェントが移動する度にプログラムコードを転送するため，移動にかかるオーバーヘッドが大きいという問題がある．

3 提案方式

提案方式では，第2節で述べた「一括して転送する方式」にキャッシング技術を組み合わせることで非常時

接続環境での優位性を確保したままモバイルエージェントの移動の高速化をはかる．

提案方式では，モバイルエージェントの移動時に，プログラムコードと一意的に対応するキーを移動先へ転送し，キーに対応するプログラムコードが移動先にキャッシュされていない場合にのみプログラムコードの転送を行い，移動先にキャッシュする．このようなプログラムコードの転送方式により，モバイルエージェントが過去に訪れたことのあるコンピュータへ移動する際には，キャッシュを利用可能となり，無駄なトラフィックを削減することができる．その結果，モバイルエージェントの高速な移動が可能となる．

4 提案方式の実装

提案方式を Maglog[2][3] 上に実装した．Maglog(Mobile AGent system on proLOG) とは，我々が開発しているモバイルエージェントフレームワークであり，モバイルエージェントの構築環境と実行環境を提供する．Maglog は Java で実装されているため，モバイルエージェントの移動手続きは，Java のオブジェクト直列化システム [4] によってエージェントの実行時状態を構成するオブジェクトをバイト配列へ変換し，HTTP/1.1 プロトコルを用いて転送することで実現した．また，プログラムコードは，XML-RPC プロトコルを用いて転送することで実現した．

5 評価

5.1 モバイルエージェントの移動に要する時間とトラフィックに関する評価

提案方式の有効性を確認するために，既存方式と提案方式について比較実験を行った．

実験内容 実験では，プログラムコードを，あらかじめコンピュータに配置しておく方式 (A) と，モバイルエージェントの移動毎に転送する方式 (B), そして提案方式 (C) について，モバイルエージェントが 2 台のコンピュータ間を 1 往復するのに要する時間と，モバイルエージェントが移動する際に発生するトラフィックを測定した．実験は，CPU=Pentium4 3.0GHz, Memory=1GB のコンピュータ 2 台を 1000BASE-T のイーサネット接続した環境で行った．また，移動させるモバイルエージェントは，実行時状態領域が 62KB, プログラムコード領域が 954KB のサイズのものを用いた．

実験結果 実験結果を図 1 に示す．モバイルエージェントの往復に要する時間について，提案方式 (C) は，1 回目の試行で最も多いが，2 回目以降の試行については方式 (B) よりも約 50%少なく，かつ方式 (A) に近い値となっている．エージェントの移動時に発生するトラフィックについては，方式 (A) は常に 64KB，方式 (B) は常に 1016KB，提案方式 (C) は 1 回目の移動では 1016KB であったが，2 回目以降の移動では 64KB となり，2 回目以降の移動では実行時状態とキーのみの転送を行っていることがわかる．

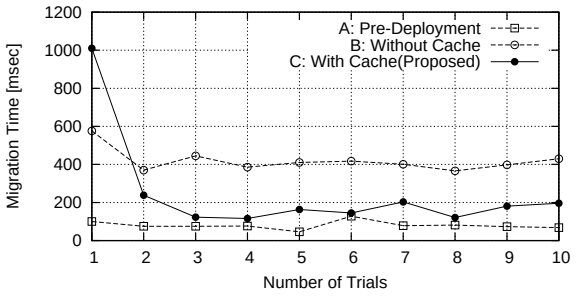

図 1: エージェントの往復移動の試行回数と移動時間

以上の結果より，エージェントが 2 台のコンピュータ間を 2 回以上往復する状況において，提案方式が有効であることが確認できた．

また，1 回しか往復しないような状況においては，提案方式 (C) がもっとも移動に時間がかかる結果となったが，実際的なアプリケーションでは，モバイルエージェントが 1 回しか訪れないようなコンピュータは，アプリケーション全体のパフォーマンスに大きな影響を与えないため問題とならないと考えられる．

5.2 実際的なアプリケーションを用いた評価

実際的なアプリケーションにおける提案方式の有効性を評価するために，我々が Maglog を用いて開発している会議日程調整システム [5] を用いた実験を行った．我々が開発している会議日程調整システムとは，モバイルエージェントが，ネットワークに接続されたコンピュータ間を移動しながら，コンピュータのユーザに対して会議に参加可能な日程の収集や交渉を積極的に行うことで，会議の開催日時を速やかに決定するためのアプリケーションである．

実験内容 実験では，プログラムコードを，モバイルエージェントの移動毎に転送する方式 (B), と提案方式 (C) について，11 台のコンピュータで構築した会議日程調整システムにおいて，会議日程調整に要する時間を測定した．実験は，CPU=Pentium4 3.0GHz, Memory=1GB のコンピュータ 11 台を 1000BASE-T のイーサネット接続した環境で行った．

実験結果 実験結果を図 2 に示す．1 回目の会議日程調整に要する時間について，方式 (B) と提案方式 (C) に差は見られなかった．ところが，2 回目の会議日程調整以降は，方式 (B) は 25 秒前後を要したのに対して，提案方式は約 12 秒で完了した．以上の結果より，提案方式の有効性が確認できたとともに，モバイルエージェントの移動を高速化することで，モバイルエージェント技術を用いたアプリケーションのパフォーマンスを改善できることを確認できた．

6 おわりに

本論文では，モバイルエージェントの移動毎に発生する，プログラムコード転送のトラフィックを削減する手法について述べた．提案方式を実装し評価した結果，モバイルエージェントが 2 台のコンピュータ間を

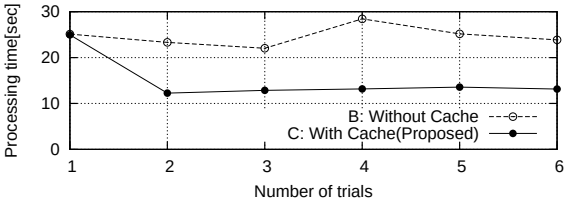

図 2: 会議日程調整に要した時間

2 回以上往復する状況において，提案方式が有効である事を確認した．また，モバイルエージェントの移動を高速化すれば，モバイルエージェント技術を用いて作られたアプリケーションのパフォーマンスを改善できることを確認した．

参考文献

[1] 一郎佐藤: モバイルエージェント技術と研究動向 (< 特集 > 情報プラットフォーム), *NII journal*, Vol. 3, pp. 53–66 (20011130).

[2] Motomura, S., Kawamura, T. and Sugahara, K.: Logic-Based Mobile Agent Framework with a Concept of "Field", *IPSJ Journal*, Vol. 47, No. 4, pp. 1230–1238 (2006).

[3] Kawamura, T., Motomura, S. and Sugahara, K.: Implementation of a Logic-based Multi Agent Framework on Java Environment, *Proceedings of IEEE International Conference on Integration of Knowledge Intensive Multi-Agent Systems* (Hexmoor, H.(ed.)), pp. 486–491 (2005). Waltham, Massachusetts, USA.

[4] Sun Microsystems, Inc.: Object Serialization, http://java.sun.com/javase/6/docs/technotes/guides/serialization/index.html.

[5] Kawamura, T., Motomura, S., Kagemoto, K. and Sugahara, K.: Meeting Arrangement System Based On Mobile Agent Technology, *Proceedings of the 2nd International Conference on Web Information Systems and Technologies*, pp. 117–120 (2006). Setubal, Portugal.